Preconditioning Kernel Matrices

K. Cutajar¹, M. A. Osborne², J. P. Cunningham¹, M. Filippone³
¹ - EURECOM, Sophia Antipolis, France
² - University of Oxford, Oxford, UK
³ - Columbia University, New York City, USA

Kernel Machines and Solving Linear Systems

- Operate in a high-dimensional, implicit feature space;
- Relate on the construction of an $n \times n$ Gram matrix $K$;

Popular kernels:
- RBF: $k(x_i, x_j) = e^{-\|x_i - x_j\|^2}$
- Matern: $k(x_i, x_j) = (1 + \sqrt{2d}) e^{-\sqrt{2d}\|x_i - x_j\|}$

- Involves the solution of linear systems $Kx = v$;
- Cholesky Decomposition:
  - $O(n^3)$ space and $O(n^3)$ time - unfeasible for large $n$.
- Conjugate Gradient (CG):
  - Numerical solution of linear systems constructed using matrix-vector multiplications;
  - $O(t^2)$ for $t$ CG iterations in practice $t = n$ (possibly worse).

- Preconditioned Conjugate Gradient (PCG):
  - Transforms linear system to be better conditioned, improving convergence;
  - Yields a new linear system of the form $P^{-1}Kx = P^{-1}v$;
  - $O(t^2)$ for $t$ PCG iterations.

Preconditioning Approaches

- Suppose we want to precondition $K_y = K + M$;
- Our choice of preconditioner should:
  - Approximate $K_y$ as closely as possible;
  - Be easy to invert.
- For low-rank preconditioners we employ the Woodbury inversion lemma:

$$K_y = U K U^T + M$$

$$P^{-1} = -I + L^{-1} + G^{-1}$$

- For other preconditioners we solve inner linear systems once again using CG.

### Formulation

<table>
<thead>
<tr>
<th>Nystrom</th>
<th>$P = K_{XX}K_{YX}^{-1}K_{YX} + M$ where $U \subseteq X$</th>
<th>Woodbury</th>
</tr>
</thead>
<tbody>
<tr>
<td>FITC</td>
<td>$P = K_{XX}K_{YX}^{-1}K_{YX} + \text{diag}(K - K_{XX}K_{YX}K_{YX}) + M$</td>
<td>Woodbury</td>
</tr>
<tr>
<td>PITC</td>
<td>$P = K_{XX}K_{YX}^{-1}K_{YX} + \text{diag}(K - K_{XX}K_{YX}K_{YX}) + M$</td>
<td>Woodbury</td>
</tr>
<tr>
<td>Spectral</td>
<td>$P = -\sum_{i=1}^n \text{diag}(r_i) + M + M$</td>
<td>Woodbury</td>
</tr>
<tr>
<td>Partial SVD</td>
<td>$K_{YY} = \Lambda\Lambda^T \Rightarrow P = A_{1:n}A_{1:n}^T + M + M$</td>
<td>Block Inverse</td>
</tr>
<tr>
<td>Block Jacobi</td>
<td>$P = \text{bdiag}(K) + M$</td>
<td>Inner CG</td>
</tr>
<tr>
<td>Regularization</td>
<td>$P = K + \delta + M$</td>
<td>Inner CG</td>
</tr>
</tbody>
</table>

### Strategy

#### Concrete Dataset

| $(n = 1038, d = 9)$ | Block Jacobi | PITC | FITC |

#### Power plant Dataset

| $(n = 9506, d = 4)$ | Block Jacobi | PITC | FITC |

#### Protein Dataset

| $(n = 45730, d = 9)$ | Block Jacobi | PITC | FITC |

### Conclusions

- Our solution:
  - Exact in the limit of iterations;
  - Straightforward to construct and easy to tune;
  - Scalable to large datasets - no need to store $K$;
  - Competitive with exact Cholesky decomposition;
  - Superior to approximate methods.
- Ongoing work:
  - Extending this work to other kernel functions and models;
  - Implementation on a distributed framework;
  - Exploiting PCG in the solution of $f(K)x = v$. 

Motivating Example - Gaussian Processes

- Marginal likelihood:
  - $\log p(y|\varphi) = -\frac{1}{2} \log |K_y| - \frac{1}{2} y^T K_y^{-1} y + \text{const.}$
- Derivatives w.r.t. $\varphi$:
  - $\frac{\partial \log p(y|\varphi)}{\partial \varphi} = -\frac{1}{2} \text{Tr} \left( K_y^{-1} \frac{\partial K}{\partial \varphi} \right) + \frac{1}{2} y^T K_y^{-1} \frac{\partial K}{\partial \varphi} y$
- Stochastic estimate of the trace - assume $E[r^2] = I$ then:
  - $\text{Tr} \left( K_y^{-1} \frac{\partial K}{\partial \varphi} \right) = \text{Tr} \left( K_y^{-1} \frac{\partial K}{\partial \varphi} E[r^2] \right) = E \left[ r^T K_y^{-1} \frac{\partial K}{\partial \varphi} r \right] = \frac{1}{N} \sum_{i=1}^N r_i^T K_y^{-1} \frac{\partial K}{\partial \varphi} r_i$
- Linear systems only!
- Laplace approximation for non-Gaussian likelihoods may be formulated in a similar way!

Experimental Setup and Results

- Exact gradient-based optimization using Cholesky decomposition (CHOL);
- Stochastic gradient-based optimization using ADAGRAD - using CG and PCG;
- GP Approximations:
  - Variational learning of inducing variables (VAR);
  - Fully Independent Training Conditional (FITC);
  - Partially Independent Training Conditional (PITC).

### Classification

<table>
<thead>
<tr>
<th></th>
<th>Spambase Dataset $(n = 4601, d = 57)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Regression

<table>
<thead>
<tr>
<th></th>
<th>Power plant Dataset $(n = 9506, d = 4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Protein Dataset

<table>
<thead>
<tr>
<th></th>
<th>$(n = 45730, d = 9)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4: Comparison of preconditioners for different settings of kernel parameters across multiple datasets. Top: Number of iterations required to solve the corresponding linear system using CG. Bottom: Rate of improvement (blue) or degradation (red) achieved by using PCG to solve the same linear system. 

Fig. 5: Error and negative log likelihood on $\gamma$ held out test data over time. Curves are averaged over multiple repetitions.